

# Technical Platform Product Management

The Key to Platform Engineering Success and Adoption



# Agenda

- ❖ Why Platform Engineering Needs Product Management
- ❖ Platform-as-a-Product
- ❖ Driving Business Outcomes
- ❖ Platform Value Model
- ❖ Platform Engineering in Practice
- ❖ Scaling Platform Engineering Adoption

# Recap - Video # 1

- Challenges and past trends that led to today's Platform Engineering
- Evolving PE in organizations
- Platform Engineering: Career Paths and Growth Opportunities
- Making it all stick
- The current and future trends in Platform Engineering
- How this book will help you?

# TPM

"Technical Product Management is the linchpin for successful platform engineering adoption and scalability."

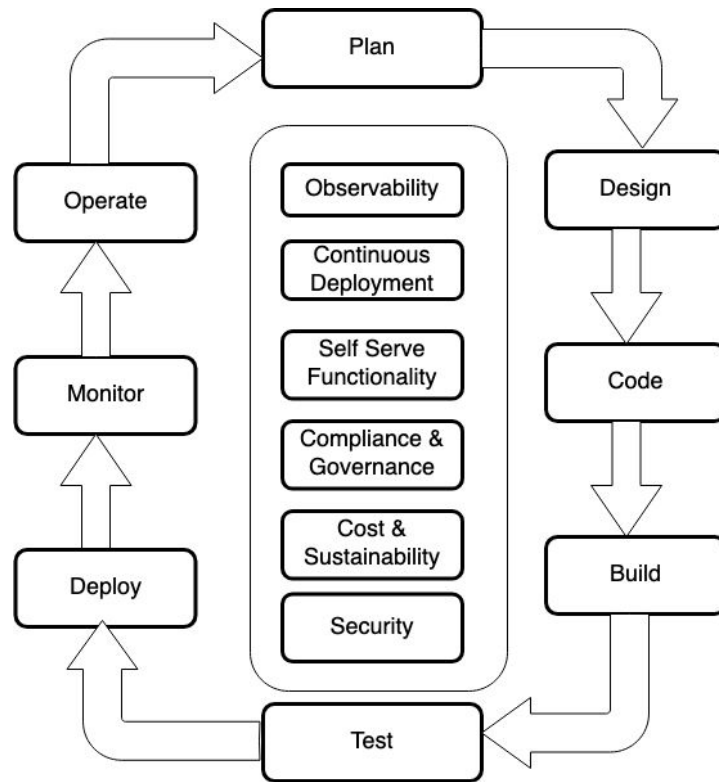


# Why Platform Engineering Needs Product Management?

# Platform Product Model

01	<b>Accelerated Business Delivery</b> Faster product evolution through increased experimentation and reduced effort as the platform matures.
02	<b>Managed Developer Load</b> Complexity abstracted into the platform, allowing developers to focus on business needs with flexibility.
03	<b>Composable &amp; Responsive Platform</b> Composable capabilities enable quick responses to new challenges and efficient experimentation.
04	<b>Security &amp; Compliance</b> Built-in guardrails ensure secure, compliant delivery from the start.
05	<b>Independent Capabilities</b> Decoupled services with separate roadmaps, avoiding dependency on the platform team.
06	<b>Faster Time to Value</b> Fewer friction points unlock rapid, high-quality delivery.
07	<b>Reduced Toil, Higher Productivity</b> Developer productivity rises with platform-as-a-service backed by clear SLAs and SLOs.

# A high-level view of the SDLC in the context of platform principles



# Why isn't DevOps scaling?

DevOps involves a significant amount of change, and change can be difficult to manage. Teams may struggle to adopt new tools, processes, and ways of working, leading to resistance or incomplete implementation.

**DevOps Team  
supporting  
Engineering Teams**

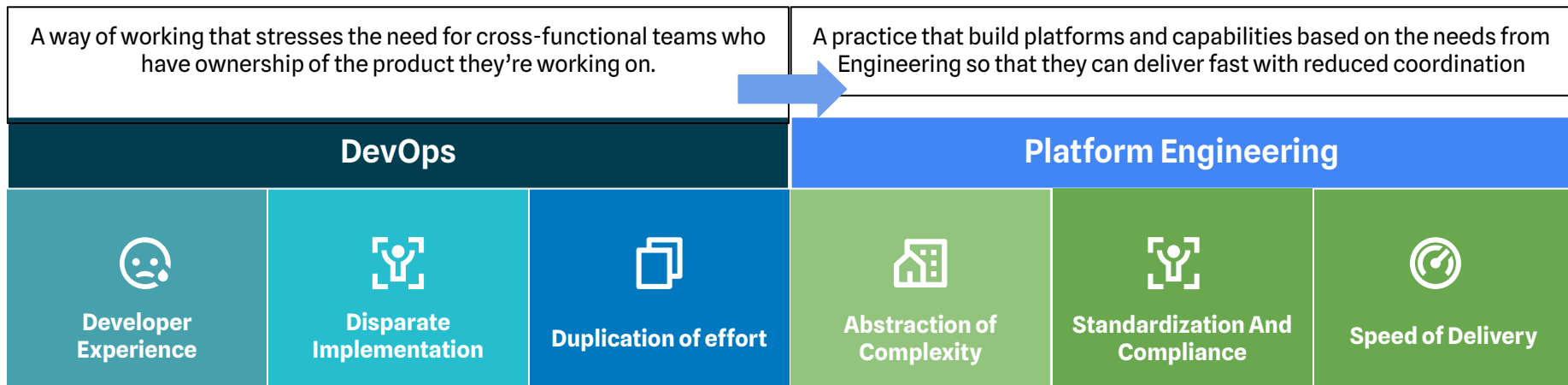
**Dependency on  
Operations and  
Approval Process**

**Inadequate  
monitoring and  
logging**

**Inconsistent use of  
frameworks,  
processes and  
automation**



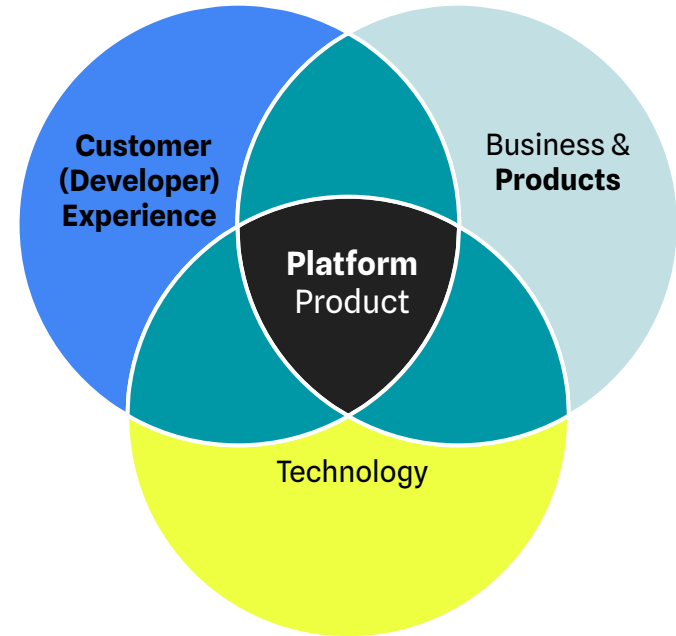
# How does Platform Engineering enable DevOps?






# Platform-as-a-Product

# Applying product thinking to platforms

- Deliver Value
- Reduce Friction
- Abstract Complexity
- Learn Rapidly



# What makes a successful Platform Product?

 <b>Vision &amp; Strategy</b>	 <b>Alignment to Business Outcomes</b>	 <b>Onboarding Experience</b>
Taking a customer centric approach	Platform vision aligned to enable business vision	Self service onboarding, changes and off-boarding
Discoverability and self service adoption	Platform and Engineering Leadership aligned on platform vision and roadmap	Optimized Customer(developer) journey
Ease of enablement and adoption	Value proposition for each platform capability enabling a business outcome	Self Service Access management
Clear value proposition, quantification and tracking	Use cases and benefits for each platform capability	Feedback loops for identifying bottlenecks and solutions
Metrics to measure and track success	Clarity on outcomes delivered for each platform capability	
Scaleable support and shared responsibility model		

# What's Unique about Platform Products

	Business Domain Product	Platform Product
<b>User Research</b>	Typically External	Always Internal
<b>Technology impact</b>	Low, delivers business capabilities	High, delivers technical capabilities
<b>Stakeholders</b>	External Customers	Internal customers, need to collaborate on roadmap, enablement and adoption
<b>Success Metrics</b>	Easy to identify and track	Contextual, difficult to track
<b>Measuring Value</b>	Clear in most cases - correlates with revenue, acquisition, retention and so on	Challenging - Tracking startup time, adoption, reuse, uptime and so on
<b>Value Delivery</b>	Easy to project, limited dependencies	Depends on Org culture, skillsets and enablement
<b>Prioritization</b>	Objective in most cases	Difficult to make it objective (depends on the level of reuse, adoption, enablement...)
<b>Support</b>	Typical application support	Needs shared responsibility model

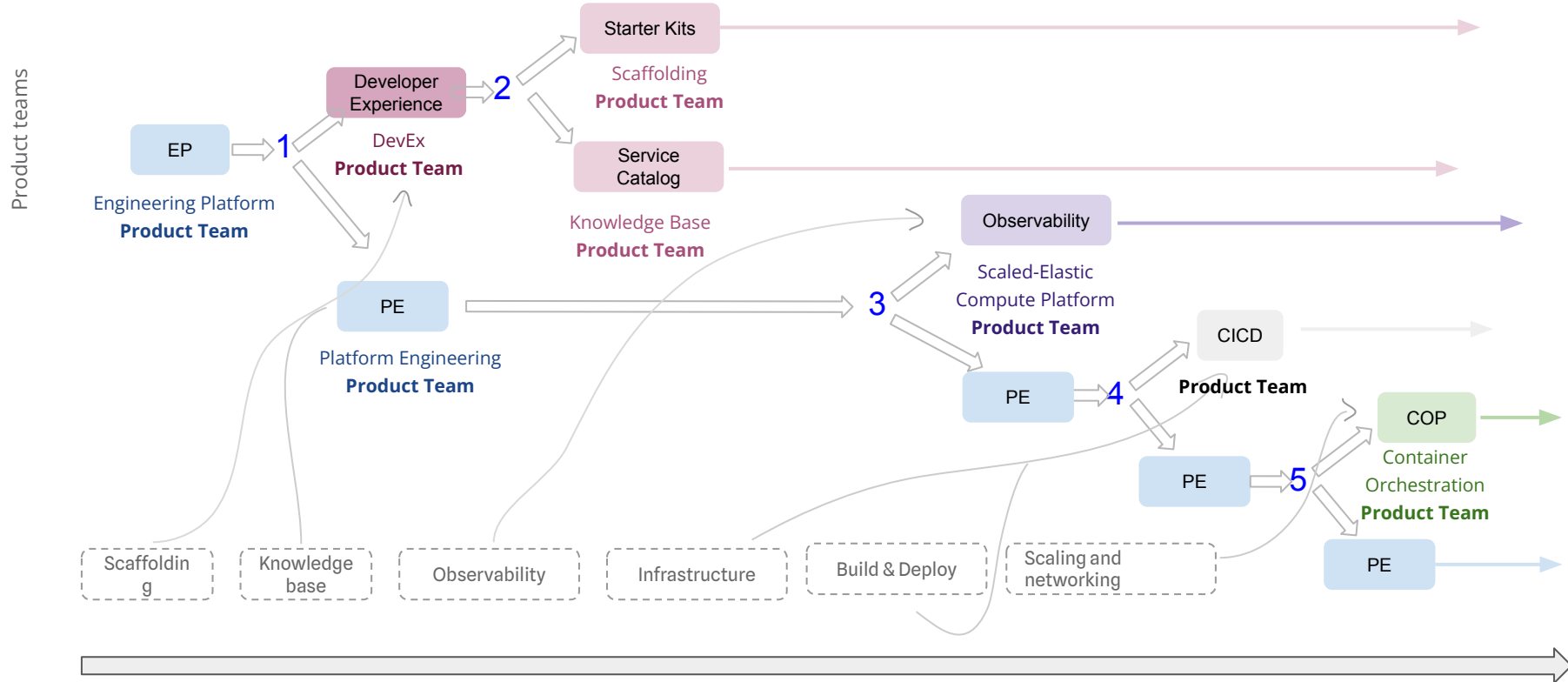
# Examples of Platform Products

A Non Exhaustive list of Platform Products and Capabilities we build at our clients

Infrastructure Related	Pipeline/Governance Related	Reporting Related	Paved Roads
IaC templates	CI/CD	Observability Platform	Vendor specific RIs
Container Orchestration	Quality Gates	Workflow Provenance	Vendor agnostic SKs
Config as Code	Shift Left Compliance	Low cost monitoring	Contract testing APIs
Service Mesh**	0 Trust Networking	Open SLO	
Multi Region Failover / DR	Policy as Code	Alerting	
Rollout tooling**	Resiliency Platforms	Traceability	
Golden Image updaters**	Secrets Management		
Scaling			

\*\*These can be built at subsets/capabilities of larger Platform Products

# Evolution of Platform Engineering Team



# What would a Platform Product Manager do?

- Focus on **technology, developers and architects**.
- Measured by Developer Experience (DevEx) **focus on the customer and business domain**
- Platform Product manager owns the delivery of a compelling Platform product.

## CUSTOMER FEEDBACK

### User Centricity

Partner with Dev teams identify patterns and design capabilities that help teams move faster by reducing cognitive load

## PSHAPE THE PRODUCT

### Enablement & Adoption

Maximize value delivery by partnering with Engineering Leadership to drive continuous enablement needed for adoption

## COORDINATE

### Shared Responsibility Model

With shared responsibility approach, support models need designed and executed effectively in for a smoother scaling across entire organization

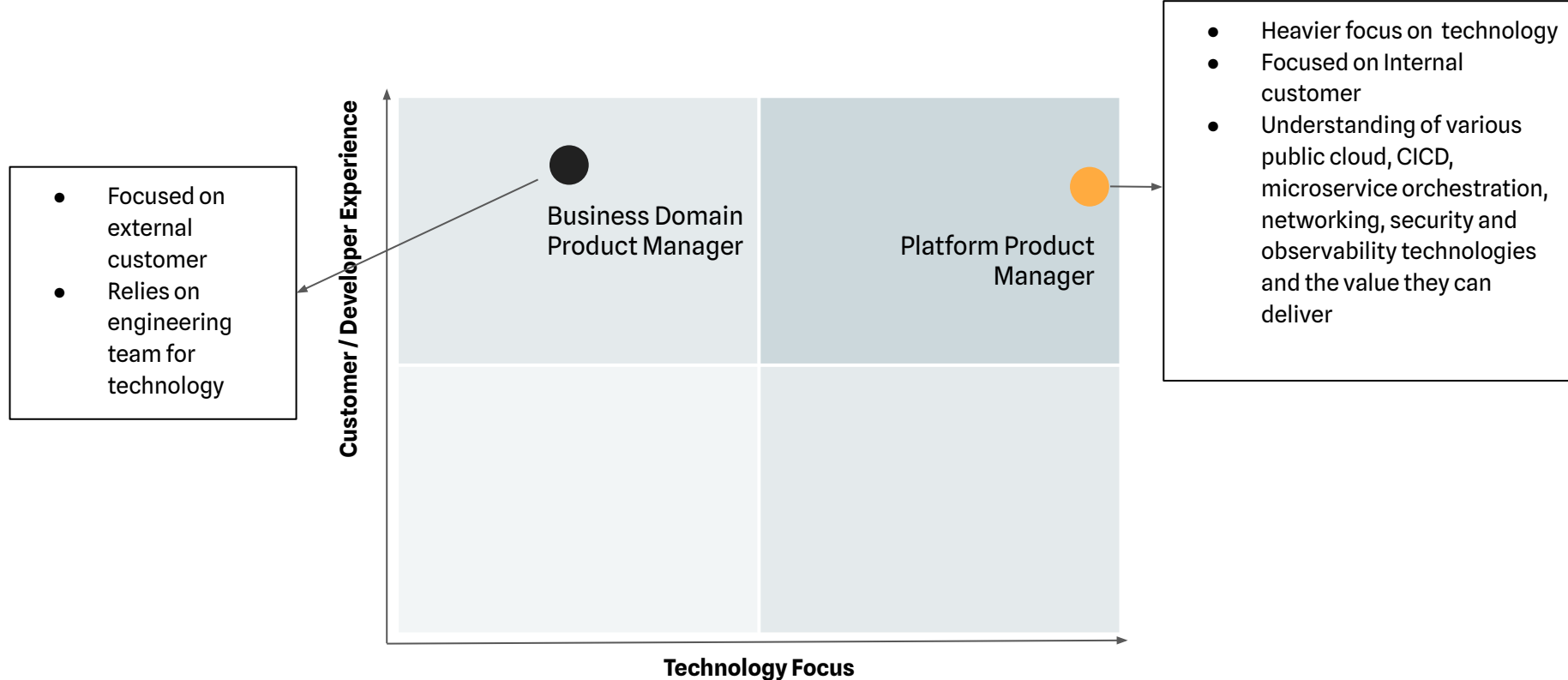
## QUALITY OF SERVICE

### SLAs & SLOs

Considering the blast radius, SLA's & SLO's need to be well defined and operationalized while accounting for public cloud and other SaaS services

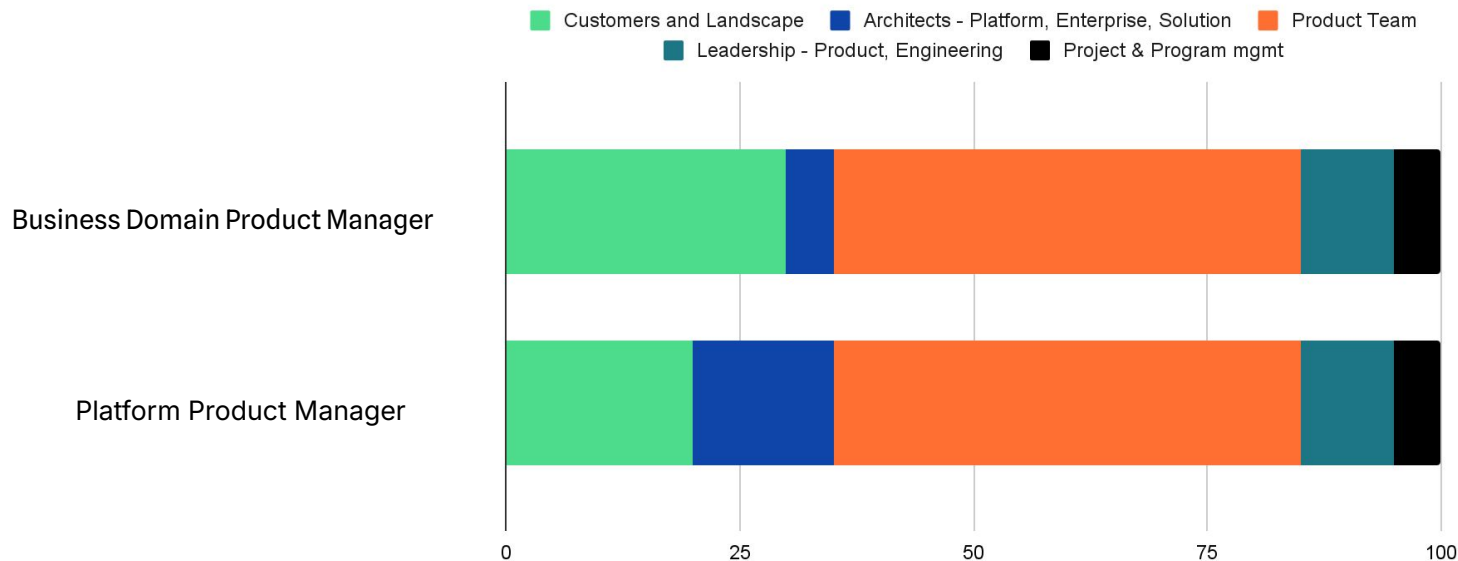


# Product Role - larger focus on technology



# Working with stakeholders

Engagement and the time spent with different stakeholders significantly vary for a Platform product compared to a Business Domain product



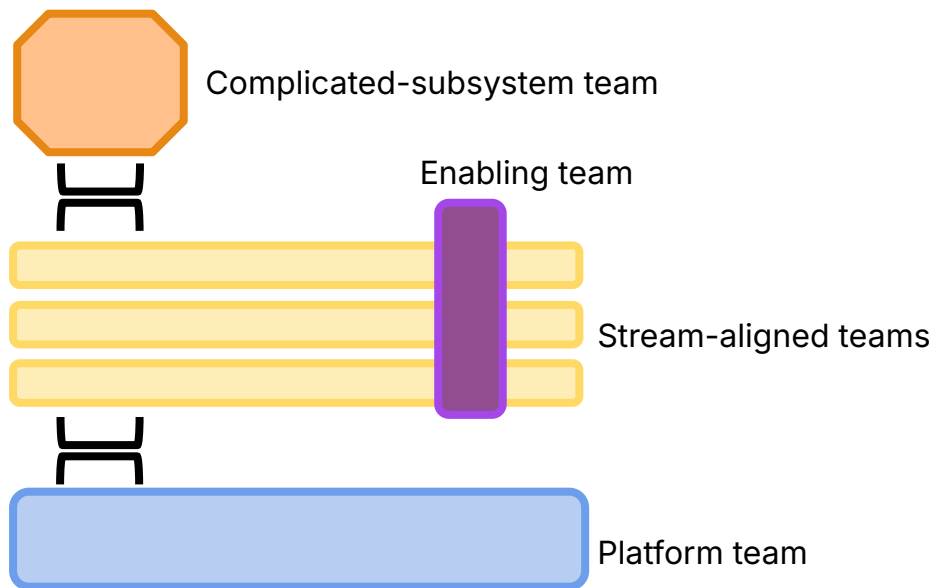
# Skills Needed as a Platform product manager

	Technical Product Manager	Platform Product Manager
Product Strategy	<ul style="list-style-type: none"><li>• Understanding of the technology need</li><li>• Design onboarding and scaling experience</li><li>• Define and measure success metrics</li></ul>	<ul style="list-style-type: none"><li>• Understanding of Developer journey and the delivery ecosystem</li><li>• Partner with Engineering, Architecture and Platform leadership to drive the design</li><li>• Ability to build an ecosystem with templates, starter kits and reference implementations</li><li>• Articulate value and drive value delivery by evangelizing and drive adoption</li></ul>
Technology Architecture	<ul style="list-style-type: none"><li>• Ability to understand and influence architectural direction for the product team</li><li>• Understanding of cloud resources and their implications on cost and CFRs</li><li>• Ability to drive API design decisions</li><li>• Ability to drive CFRs such as scalability, maintainability, high availability and user access</li></ul>	<ul style="list-style-type: none"><li>• Ability to understand customer's (engineering teams) architectural direction in order to drive PE product definition, roadmap and direction</li><li>• Understanding of Cloud Native, Multi-cloud, IaC, CaC patterns</li><li>• Ability to drive CFRs such as scalability, maintainability, high availability, user access, multi tenancy and security <b>as Enabler</b></li><li>• Understanding of networking patterns such as VPC/VPN, subnets, gateways, DNS and other</li></ul>
Tech Stacks	<ul style="list-style-type: none"><li>• APIs, Messaging, Fullstack technologies,</li><li>• CI/CD, IaC, Observability as a Consumer</li></ul>	<ul style="list-style-type: none"><li>• Native, Hybrid, Multi-Cloud patterns</li><li>• CI/CD, IaC, CaC, Observability <b>as Producer</b></li><li>• Container Orchestration</li></ul>
User Experience	<ul style="list-style-type: none"><li>• Ability to drive self service design</li><li>• Manage API performance metrics</li></ul>	<ul style="list-style-type: none"><li>• Ability to drive self service, composable and extensible design to remove friction</li><li>• Ability to drive security and compliance design</li><li>• Ability to drive enablement and seamless onboarding</li></ul>

*\*Not an exhaustive list - For representation only*

# Translating to Measurable Business Outcomes

# Where **do** Platforms fit in the Organization?



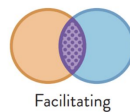
Collaboration

Two teams work together on a shared goal, particularly during discovery of new technology or approaches



X-as-a-Service

One team consumes something provided by another team (such as an API, a tool, or a full software product)



Facilitating

One team (usually an enabling team) facilitates another team in learning or adopting a new approach.

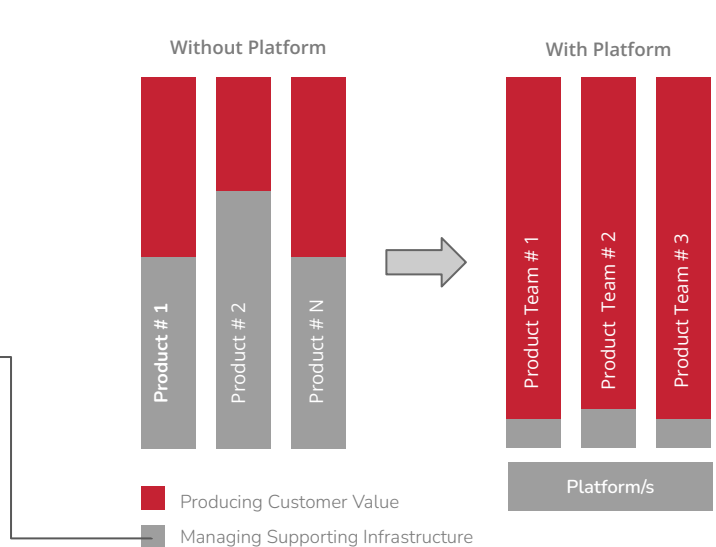


## Platform Teams

- **Simplify otherwise complex technology** for Stream-aligned teams
- **Collaborate closely** with Stream-aligned teams for faster feedback
- **Lead by example** by consuming platform capabilities internally
- **Minimize the extraneous cognitive load** and maximize Germane cognitive load for Stream-aligned teams
- Operate in **x-as-a-service** interaction mode while occasionally operating in *collaboration* mode

# Impact of Platform Team

- Provision and scale infrastructure
- CI/CD pipeline
- Observability - Application, Infra, CI/CD....
- Standards and Governance - Cloud, On-Prem
- Networking with other applications and resources
- Security - Policy management
- Secrets management
- Certificate management
- Identity management
- Scaffolding - starter kits
- Service catalog
- Knowledge management
- Cloud cost management



*By capturing common activities all teams need to do to deploy and operate their application in production, platform capabilities allow those teams to spend more of their time on value-added activity.*

# Metrics to Consider

Like value, measuring success is another challenging area for Platforms. Typical metrics like cost, revenue, ses duration cannot be directly measured for a Platform product.

Metric	Product/Capability Examples	Type
Accelerate Metrics	IaC modules, pipelines, Cert management, API management, capabilities that enable cloud adoption or migration	One of the components
Adoption rates	IaC modules, pipelines, Cert management, API management, capabilities that enable cloud adoption or migration	Direct
Number of incidents averted	API management, Cert management, Key Vault solutions	Direct
Number of workloads deployed	Container orchestration platform	Direct
Number of Support tickets	Container orchestration platform	Direct
Execution time (deployment/build)	Build agents or containers	Direct
Availability	Container orchestration platform, API Management, Cert Management	Direct
Reuse	CICD pipelines, connectors, integrations	Direct
CSAT, NPS (Net Promoter Score)	All	Direct

# Aligning with Business Outcomes

Meet the adoption and value goals by aligning with business outcomes. PE teams would need a **Product Catalog** to drive the awareness and understanding among customer teams to influence the prioritization and adoption

	Time to Market 	Quality & Reliability 	Predict, Prevent & Recover fast 	Enable Innovation 	Cost Efficiency 
Capability 1	<input checked="" type="checkbox"/>			<input checked="" type="checkbox"/>	
Capability 2		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		
Capability 3	<input checked="" type="checkbox"/>				<input checked="" type="checkbox"/>
Capability n		<input checked="" type="checkbox"/>			



# Platform Value Model

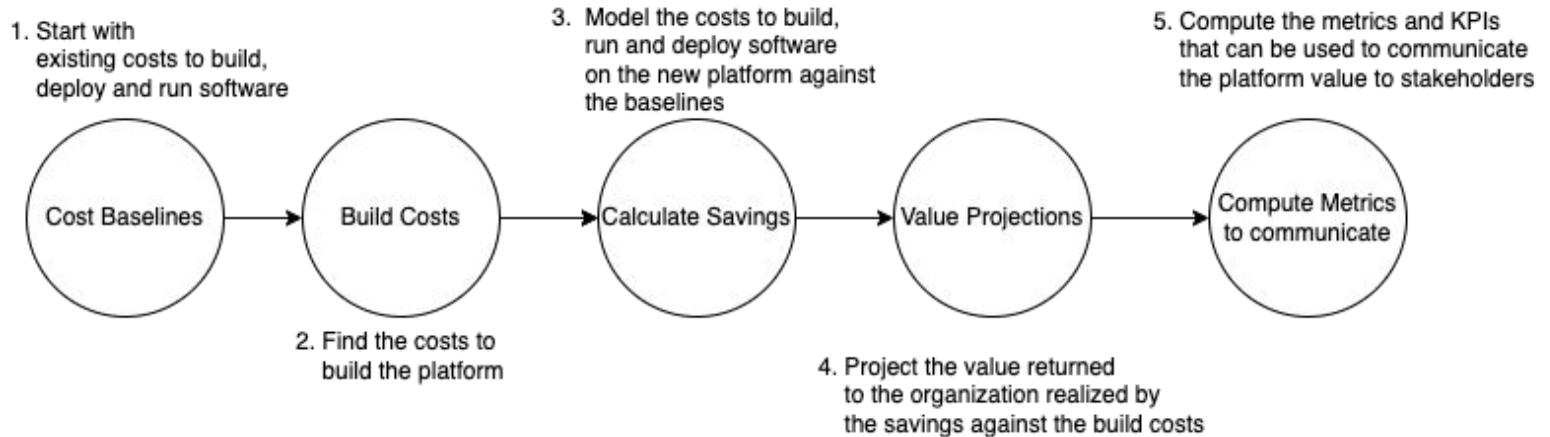
# Are platforms delivering value?

Tracking the value generated from platform investments is important because it allows organizations to determine the ROI of their investments, allocate resources effectively, identify areas for improvement, tweak the strategy improve performance. This information helps organizations to focus on areas that generate the most value and to achieve their business objectives more effectively.

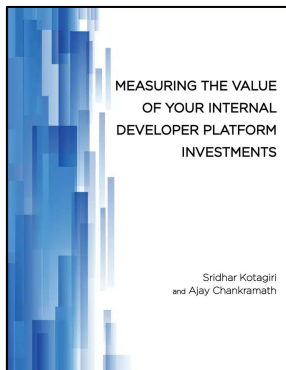
Patterns	Anti-Patterns
<ul style="list-style-type: none"><li>• Higher adoption rate - A successful platform will make the developer's job easier, more efficient, and more effective</li><li>• Improved accelerate metrics</li><li>• Decrease in development costs, such as developer efficiency, license and infrastructure costs</li><li>• Higher developer satisfaction with easier and more efficient ways to do their job</li></ul>	<ul style="list-style-type: none"><li>• Low adoption, platform needs a mandate from leadership to use</li><li>• Adoption metric may look good due to mandate but not demonstrate value</li><li>• Reliance on lagging metrics, it may be too late to make any course correction</li><li>• Resistance to change, indicates the lack of buy-in from the users and other key stakeholders</li></ul>

# Value Modeling Lifecycle

- Product and Engineering leadership is always looking for ways to justify and prioritize the product investments
- Unlike business domain products, it is quite **challenging to quantify the value for Platform Engineering Products**.



# Download the Value Model and Test it out yourself!



[Value Model](#)

# Value Modeling Demo Use Case



**Business Context:** ABC corporation is experiencing exponential growth. They are planning to adopt microservice architecture to support the growth and deliver business capabilities faster



**Problem Statement:** Product engineering teams are at various levels of enablement. Some containerize their workloads and are able to provision their own cloud infrastructure and others rely on operations team to do it for them.



**Opportunity:** Platform engineering team is considering building a container orchestration platform to help reduce the cognitive load for their customer teams and abstract away the complexity of building and deploying their own infrastructure



**Solution:** Platform architecture already evaluated a number of solutions and ended up choosing AKS as the solution for building container orchestration platform

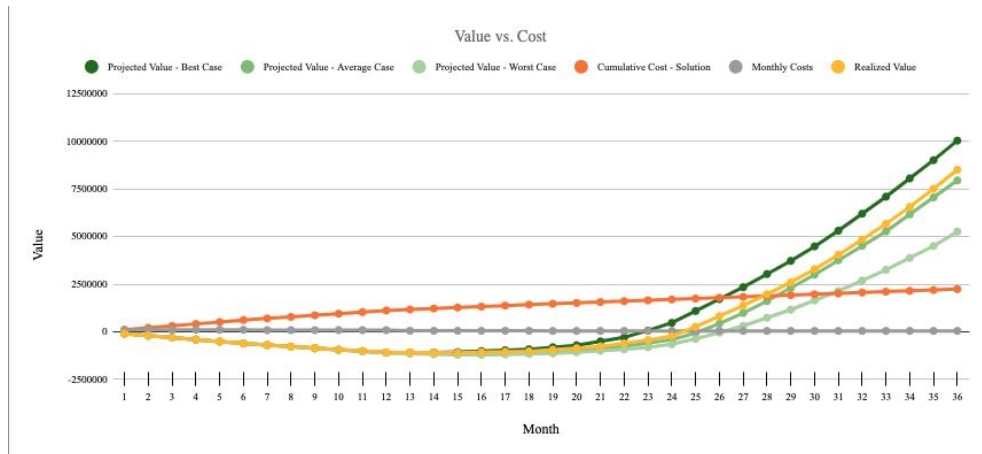


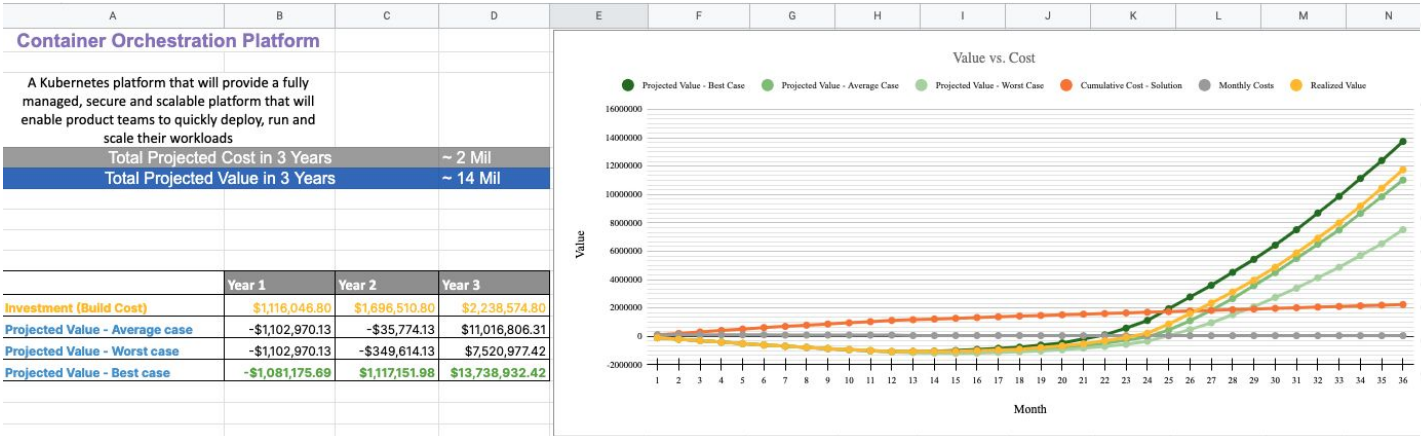
**Investment range:** \$1 to \$3M  
**Value expected:** \$5 to \$10M



**Pre-requisites:**

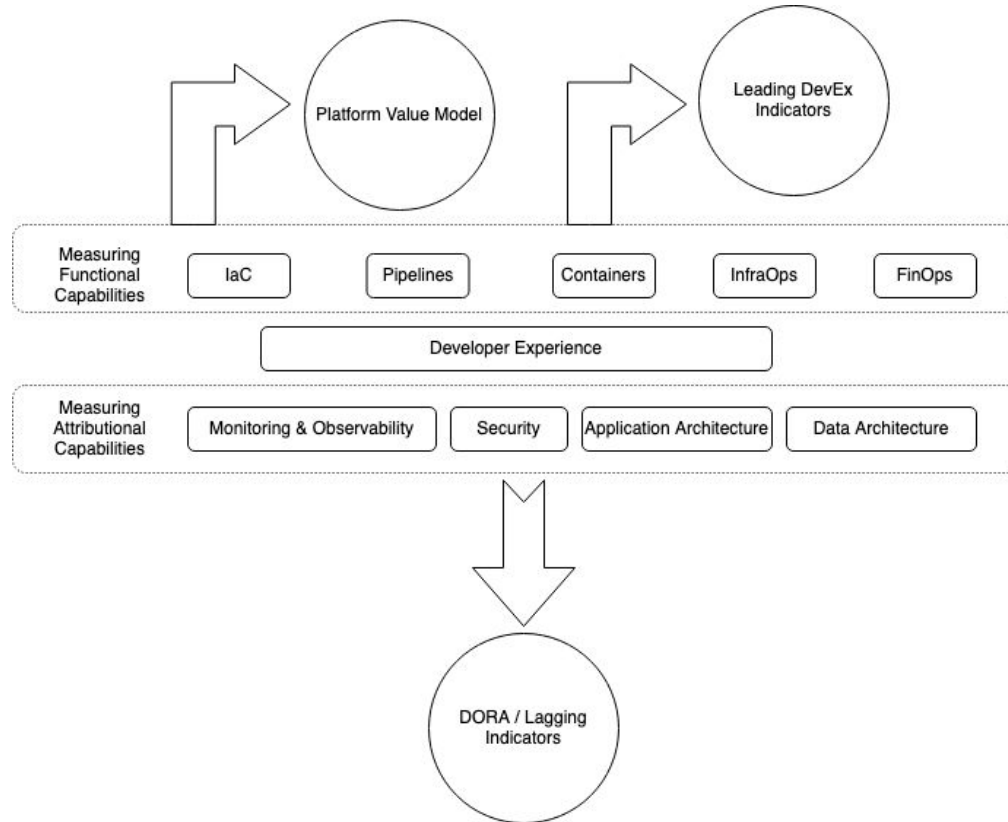
1. Team size required
2. Infra and licencing needs
3. Measurement criteria
4. Timeline for realizing the value





# Platform Product in Practice

# Capability Model





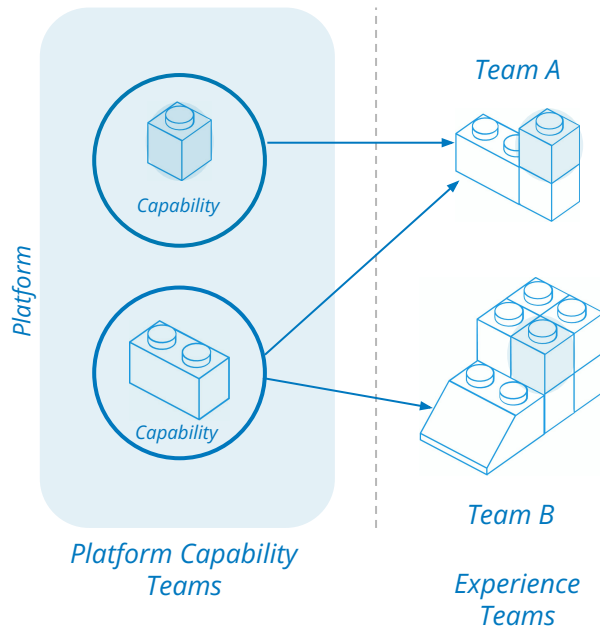
# Composable Design

## Capabilities needed to leverage opportunities

*Treating development teams as customers provides scale for innovation.*

By focusing on developers as customers and building reusable capabilities for them, platform teams accelerate the development of innovation at the experience level.

By following the unix philosophy and treating platform capabilities as Lego blocks that do one thing and do it well, teams can mix and match Lego blocks to get their job done.



# Opportunity Qualification

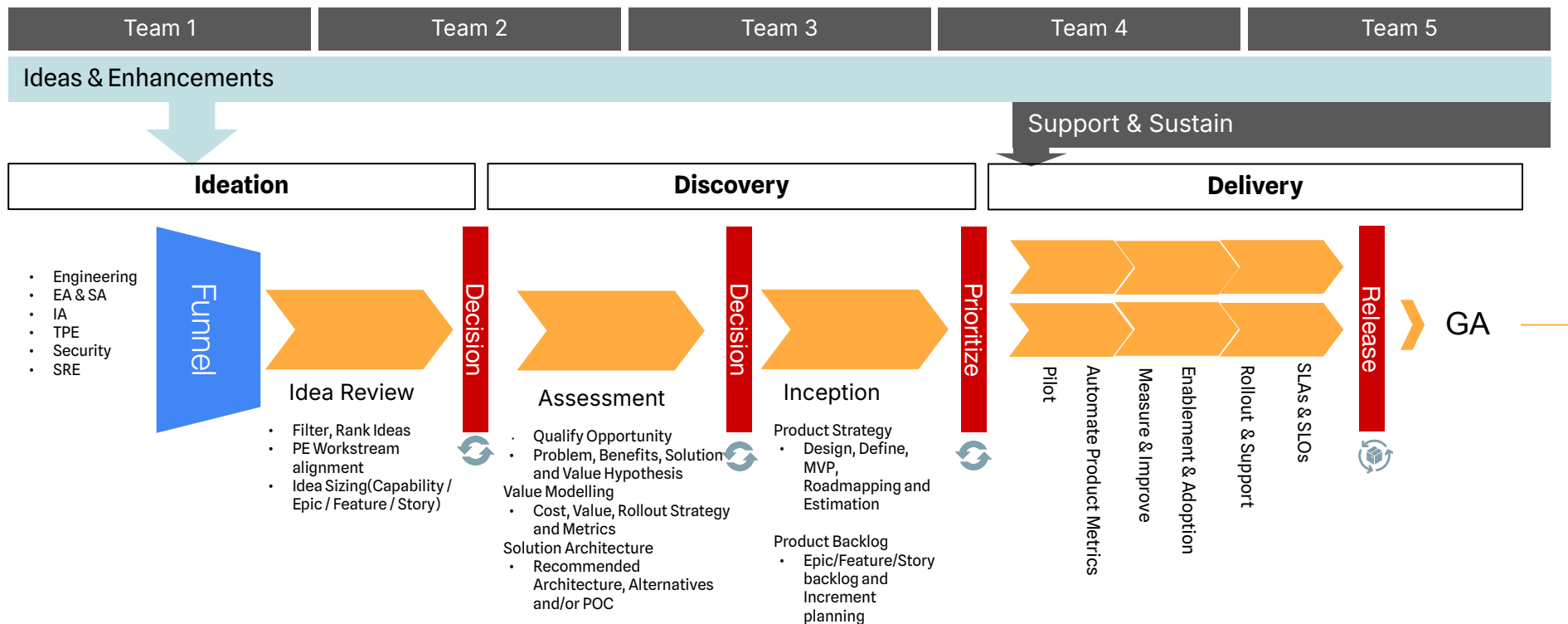
1. Adoption Potential
2. Strategic Value Alignment
3. Design & Architectural Quality
4. Business Context Independence
5. Longevity & Maintainability
6. Ownership and Operational Fit



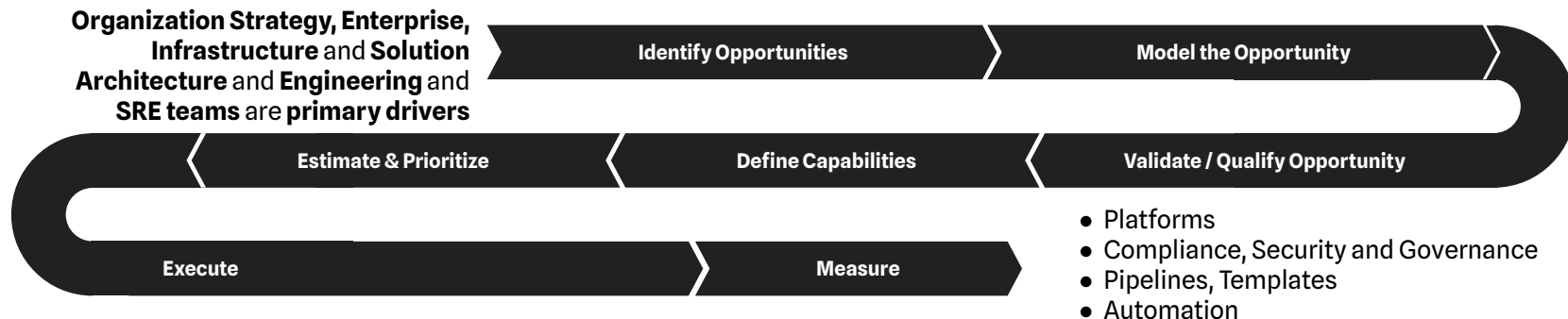
# Platform Intake

*Product Intake Workflow applied to Platform Engineering*

Customer Facing Products
Business Platform Products
Technical Platform Products
IT Security
SRE



# Platform Development Life Cycle



Thin slicing, use of right tools, frameworks, communication are critical for better visibility and incremental value delivery

Right level of partnership with Product and Engineering teams is key in building the right platform and driving the adoption

# Product management and delivery

## Ideation

What should we build?  
(develop insights, strategy and concepts)

- What problem are we solving?
- Cost benefit analysis — What is the value or solving this problem?
- Market sizing — What is the market size?
- User research — Who are the customers? What are the customer needs?
- Competitor analysis — Who are our competitors?
- What is the value proposition and our competitive differentiation/advantage?
- What is the business model?
- What are the industry trends?
- What is our vision and strategy?

## Discovery

How should it be built?  
(define, develop and launch)

- Prioritise product requirements — work breakdown structure (features / epics / stories)
- Solution architecture and alternatives evaluation
- Manage scope and trade-offs
- Roadmapping — product planning
- Operations planning and preparation
- Change management
- User interface and user experience
- Lean-agile development
- Usability testing
- Product launch plan (alpha, beta, etc)
- Customer communication
- User training / education
- Shared Responsibility Model

- Needs additional focus for Platform

## Delivery

Evaluate, optimise and promote

- Evaluate metrics
- Validate value
- Enablement, Pairing, Lunch & Learns
- Evaluate performance against SLAs/SLOs
- Analyse P&L
- Reassess competitor landscape
- Go-to-market plan
- Marketing mix & PR
- Pricing & Promotions
- Collateral
- Sales enablement & support training

- Needs lower focus for Platform

# Shared Responsibility Model

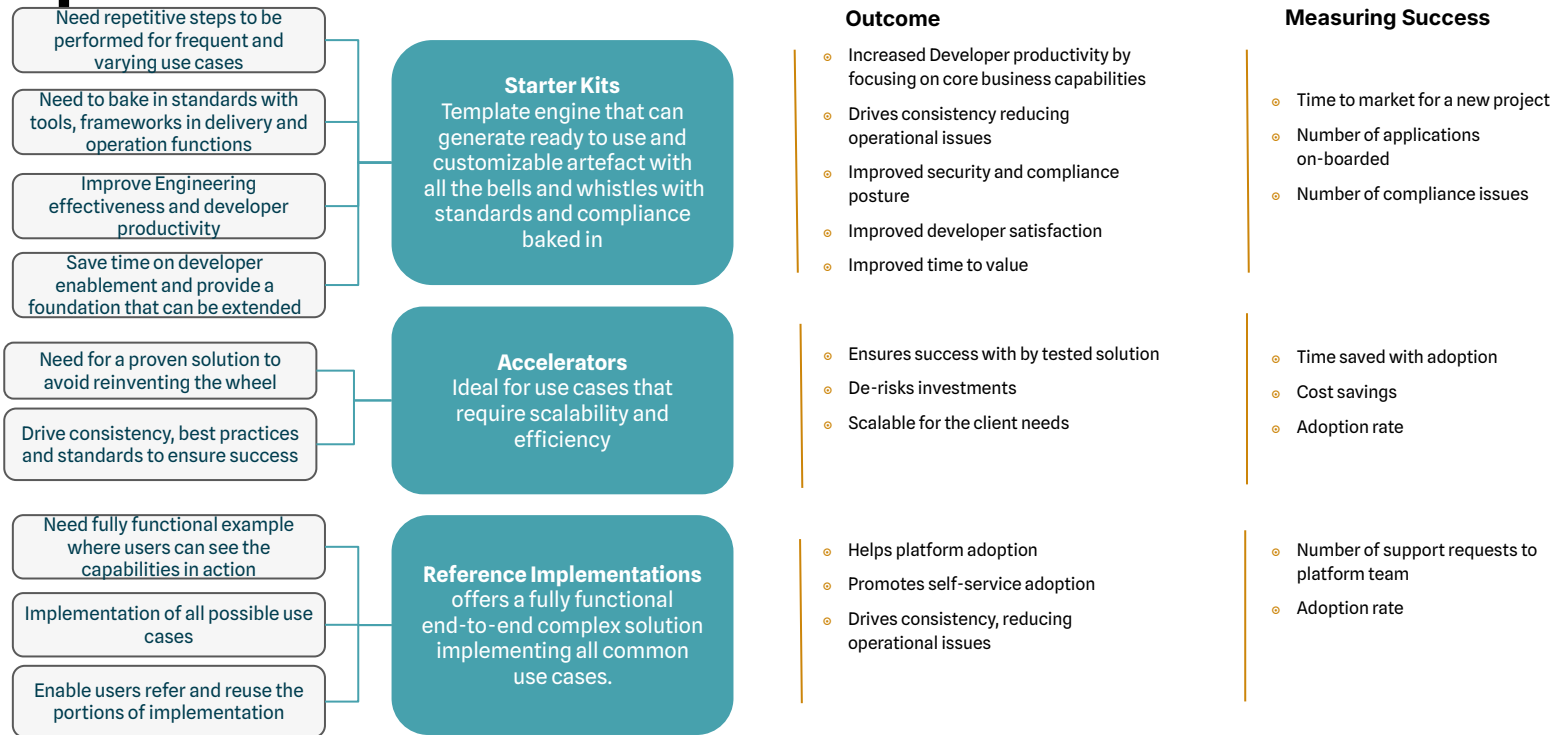
While Platform Engineering teams take ownership of the platforms and capabilities, the responsibility is shared with product engineering teams consuming platform capabilities. PE teams assume responsibility for the **resilience and availability of the platform** while customers take the responsibility for the pipelines, workloads and other **components running on the platform**.

PE team needs to ensure a **clear responsibility matrix** is documented and published while providing required level of access, isolation and observability for the customers to remove friction

										Provider	PE Team		App Team	
Observability and Alerts	Build & Deployment Tasks		Application Security		Service-to-Service Communication		Database Migration & Upgrade		Handle Outage and Restore			Troubleshooting and Debugging		
	Pipeline Orchestration	AuthN/AuthZ Config	Ingress Configuration	Blue/Green & Canary	Zero-Downtime Deploy	Application Fault Tolerance	Resiliency Testing							
	Executors	Identity Service	Service Mesh		Node Scaling, Fault Tolerance		Upgrades & Maintenance	Multi-Region Failover & DR						
	CI/CD Tooling		Container Orchestration Platform											
	Cloud Infrastructure & SaaS Tooling													

# Scaling Platform Engineering


# Role of Starter Kits, Accelerators and Ref. Implementations





# Paved Paths

Not all applications are the same, but with consistent architectural patterns and deployment practices, a paved path can be provided to make getting code to production easy from day 1



**Deploy to  
production right  
away**

**Ensure best  
practices and  
compliance**

**Extend and  
customize  
deployments**

**Diverge from the  
paved road in part  
or whole**

# Ways to Drive Adoption?

The concept of platform adoption has to be driven by the value it provides. The basic idea is that if the capabilities indeed reduce the cognitive load on the developers, it will have a sustained usage

## Voluntary

Capabilities impacting 4 key metrics, developer efficiency, reducing toil, developer and operational experience

- Capabilities that make dev's job a lot easier, compelling them to adopt
- Easy onboarding to the platform, reducing any cognitive load on devs
- Autonomy and flexibility to use platform capabilities as it fits their needs
- Make stakeholders part of setting the vision and strategy

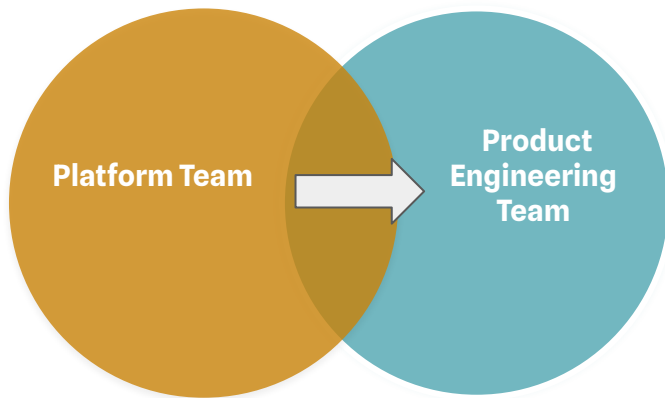
## Mandatory

Capabilities impacting security, governance, observability, compliance and cost

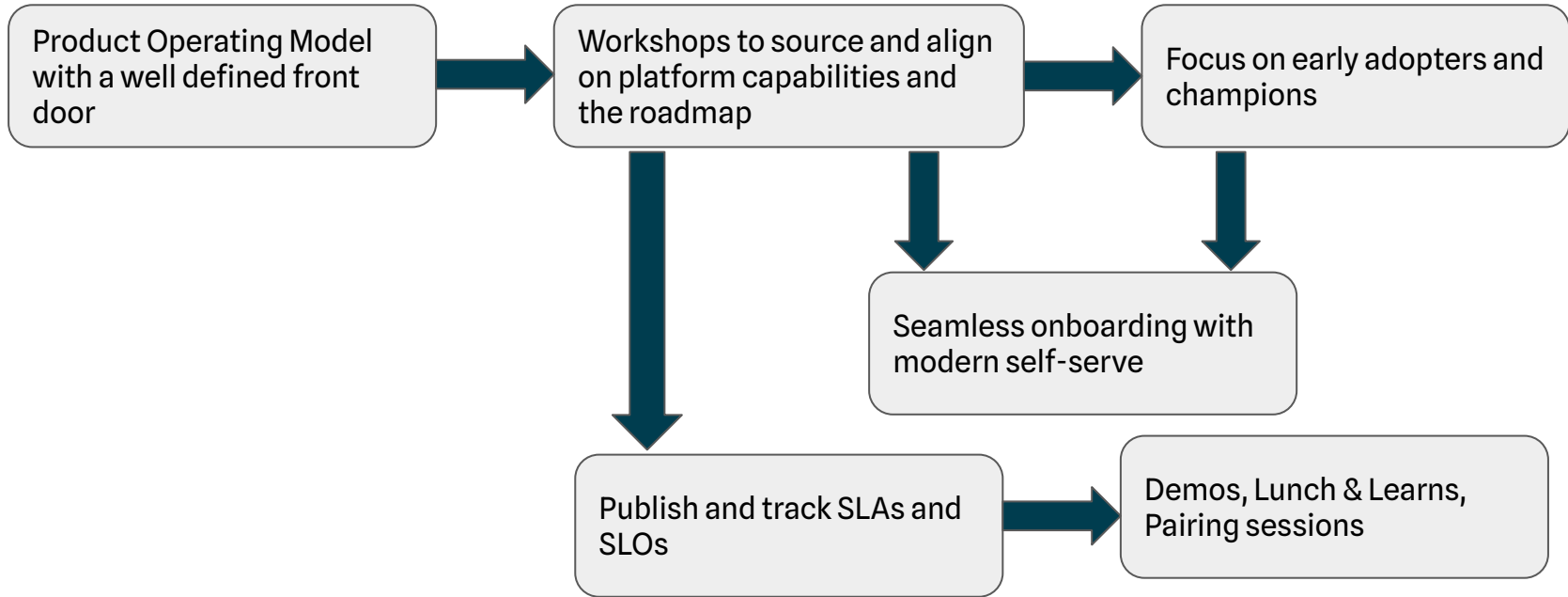
- Security compliance requirements baked into CI/CD pipelines
- Automated approval workflow build into CI/CD pipelines
- IaC and other automation addressing governance requirements, enabling consistency and compliance
- Automated instrumentation, templates and tooling for consistent Observability

# Adoption Strategy

Adoption is one of the biggest challenges for tech/business capability platforms. Lack of trust, clarity, buy-in, enablement and prioritization can lead to low adoption rates and hence lower value realization




# Adoption Acceleration Strategy



# Platform of Platforms - Adoption through a Platform

An overarching Platform that enables Self Serve Capability and tracks maturity and value realization for engineering platforms

**SUSTAIN** 

Support

- Debug & Trace
- Request Support
- Request Feature
- Launch and support communication

Adoption Metrics

- Number of users reached
- Value generated
- Metrics by customer
- SLA and Error budget status



## EXPLORE

- Platform Catalog
- Platform Metadata
- Map Platform Capabilities
- Value Proposition
- Purpose and Use cases
- Maturity Posture
- SLAs and Error Budgets
- Templates and Standards

## ONBOARD

### Knowledge

- Request Access
- Create Sandbox Environments

### Knowledge

- Customer Journeys
- Support Model
- Self Service Documentation
- Change Management
- FAQs, Lessons Learned

# Next Video - Compliance at the Point of Change

Enabling secure, compliant innovation without slowing down development



## Shift-Left on Compliance

- Introduce compliance early in the SDLC
- Detect & resolve issues before deployment



## Automated Guardrails

- Enforce policies via automation (e.g., policy-as-code)
- Prevent non-compliant changes proactively



## Developer-Centric Integration

- Embed checks into dev tools & CI/CD pipelines
- Ensure low-friction adoption by dev teams



## Context-Aware Enforcement

- Adapt compliance checks based on environment & risk
- Continuous evaluation, not just static gates

# Effective Platform Engineering

- **Effective Platform Engineering** book by [Manning](#)
- MEAP currently out awaiting print version soon



# More Information

<https://effectiveplatformengineering.com/>

## Effective Platform Engineering



Buy Now



[Ajay Chankramath](#)



[Nic Cheneweth](#)



[Bryan Oliver](#)



[Sean Alvarez](#)



"Effective Platform Engineering" is a comprehensive guide that introduces platform engineering as a discipline, focusing on creating developer platforms that enhance team efficiency and streamline application deployment. The book provides practical insights into designing and managing platforms that bridge the gap between operations and development, automating tasks throughout the software development lifecycle. Readers will learn to build internal developer platforms and portals, ensuring seamless adoption and satisfaction among teams. The authors emphasize the importance of secure, scalable Kubernetes-based engineering platforms and offer strategies for implementing effective Service Level Objectives to boost trust and adoption. Additionally, the book explores cutting-edge integrations of Generative AI tools to enhance developer productivity, providing readers with the knowledge to leverage the latest advancements in code generation.

Through practical examples and real-world scenarios, "Effective Platform Engineering" demonstrates how platform engineering differs from traditional DevOps and the unique value it brings to organizations. The book delves into both patterns and anti-patterns of platform development, guiding readers in designing and deploying secure, scalable, and observable engineering platforms. With the inclusion of diagrams, code samples, and exercises, readers can visualize key concepts and solidify their understanding. This resource is tailored for DevOps engineers familiar with Kubernetes, cloud environments, and infrastructure-as-code, aiming to equip them with the skills to establish platforms that reduce workloads, improve consistency, and accelerate software delivery.

Discover how platform engineering is revolutionizing the developer experience and operational efficiency. [Learn more](#)